WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
L3 not l4	161

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L5

Refine Search

Recall Text | Clear

Search History

DATE: Monday, November 03, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ			
L5	L3 not l4	161	L5
L4	L3 and phase?	20	L4
L3	L2 and (install$3.ti. or install$3.ab.) and (@ad<20000922 or @rlad<20000922 or @prad<20000922)	181	L3
L2	(install$3 near3 software) and (setup or integrat$3) and director$3	1946	L2
L1	(install$3 near3 software) and (setup and director$3)	1039	L1

Samuel all

END OF SEARCH HISTORY

WEST

Generate Collection Print

L4: Entry 4 of 20 File: PGPB Jan 31, 2002

DOCUMENT-IDENTIFIER: US 20020013939 A1
TITLE: Request based automation of software installation, customization and activation

Abstract Paragraph (1):
The invention relates to a system and method for request based installation, customization and activation of software products on a computer system by means of a setup infrastructure (20) for coordinating the process phases. A plurality of exploiter plug-ins (24) are provided, each assigned to a program product already installed in the computer system or a program product to be installed. The exploiter plug-ins are used by the setup infrastructure for indicating a program package description and program product description and for providing customization dialogs and support to instantiate and activate the program to be installed. Bind supporter plug-ins (23) are assigned to selected ones of the program products and used for supporting the binding procedure by providing bind services to the setup infrastructure. The supporter plug-ins are used by the exploiter plug-ins to establish binding to at least some of the program packages or products stored, where the binding parameters needed by the exploiter plug-ins are indicated to the supporter plug-ins by bind requests. Package adapter plug-ins are provided to provide data format information for enabling the setup infrastructure to identify a program product to be installed and to generate product definition data. A management directory (27) is used for storing data associated with a particular program product and references to program products in a program library of the computer system.

Priority Filing Date (1):
19991230

Summary of Invention Paragraph (6):
[0005] Large multi-user computer systems such as, for example, IBM OS/390, are sensitive about introduction of new software programs or services. These computer systems are used for many different applications. The same software programs/packages may need to be set up differently depending on what other software is installed and what applications they will be used for.

Summary of Invention Paragraph (7):
[0006] A straight forward and highly productive approach as used on workstations, where endusers usually install software packages themselves using an automated install process, is not feasible in large multi-user environments. This is mainly because typically thousands of endusers share such systems. Any failure during the installation process can jeopardize the availability or performance of the system, thereby impacting other users, or adversely impacting the protection of sensitive information belonging to the enterprise. Usually, dedicated technical staff is responsible for coordinating the use, administration and maintenance of such systems. Examples are installing and servicing of new software packages, authorizing access to sensitive or proprietary data, providing services and access from networks, organizing placement of databases, etc. Furthermore, many enterprises have multiple compute center sites. Often, the software setup process has to be repeated on each of those sites, because of slight variations in the computing environments.

Summary of Invention Paragraph (8):
[0007] Simpler software setup procedures are wanted in such complex environments. Such procedures should also enable and ensure enterprise specific responsibilities and policies.

Summary of Invention Paragraph (11):
[0009] It is an object of the invention to provide an improved software setup procedure for large computer systems.

Summary of Invention Paragraph (14):
[0012] The system according to the invention, as defined in the claims, comprises a setup infrastructure for coordinating the loading, customization, binding and activation of the program package or product to be installed. A plurality of exploiter plug-ins, each connected to one of said system resources, are used by the setup infrastructure means for creating a product definition, for providing customization dialogs and for supporting the initialization and modification of the program package or product to be installed. A plurality of supporter plug-ins are provided, each connected to selected ones of said system resources for supporting the binding procedure by providing bind services to the setup infrastructure. The supporter plug-ins are used by the exploiter plug-ins to establish binding to at least some of the program packages or products already installed, where the binding parameters needed by the exploiter plug-ins are indicated to the supporter plug-ins by Bind-Requests.

Summary of Invention Paragraph (15):
[0013] According to another aspect of the invention package adapter plug-ins are provided which identify various data formats, one of said third type plug-ins is assigned to a program product to be installed for indicating to the setup infrastructure the specific package formats as used by said program.

Summary of Invention Paragraph (16):
[0014] According to another aspect of the invention a management directory is connected to the setup infrastructure for storing all data associated with a particular software package or product and references to portions of said software package or product in a system library. These references are used by the setup infrastructure for accessing said data in a system library wherein said software package or product is stored.

Summary of Invention Paragraph (18):
[0016] Due to the declarative nature of bind requests and the capability to have a set of SI-supporters, the setup task will be much more complete. Support will not only exist to set system parameters, but there will also be supporters for areas like security, TCP/IP, etc. Therefore a program product can be taken into use after the request based software setup is complete without additional manual interaction.

Summary of Invention Paragraph (19):
[0017] Since properties are stored in the management directory, the plug-ins of the first type can provide intelligent customization dialogs. These dialogs can work with various sources of parameter values: defaults provided by the plug-ins of the first type, customization data from previous releases, primed data from the target system and customization data of other products. This can be used to achieve increased ease of use by offering a guided mode relying on defaults, further an intelligent migration support from an older to a newer program product release, an intelligent checking of prerequisites and corequisites, and an intelligent and ease of use customization by understanding the context in which a product will be used.

Summary of Invention Paragraph (20):
[0018] Since properties and bind requests can be exported and imported, and since bind requests contain references instead of copied values, installations can be replicated with minimal human interaction on the receiving side. This capability of replicating reference installation saves redundant effort for setup.

Summary of Invention Paragraph (21):
[0019] Due to the possibility of policy-based procedures, the customer's organization of authority for critical areas (security administration, database administration, network administration, etc.) are respected. If the customer changes his organization, the request based software setup can easily be adapted.

Brief Description of Drawings Paragraph (5):
[0025] FIG. 3 a functional representation of the setup infrastructure as shown in FIG. 2;

Brief Description of Drawings Paragraph (6):
[0026] FIG. 4 a functional flow diagram of the setup process; and

Detail Description Paragraph (2):
[0028] FIG. 1 relates to the prior art and shows how a typical program product install process is executed with a known workstation 8 being operated by a system control program 10 which includes a program installation, customization and bind component 11 and using a program source 12, a dialog support 13 and a directory 14. As indicated by connections 15 and 16 a program product is loaded from program source 12 into the workstation 8. The dialog support 13 provides via connection 17 program customizing information which is stored in a directory 14 through connection 18. By means of address information contained in the directory and by using the customizing information the program product is bound to the system control program of the workstation as indicated by the connections 19.

Detail Description Paragraph (4):
[0030] The request based processing according to the invention uses a different paradigm. It separates the `what is needed` from `how` and `when`. The customization process indicates via Bind Requests what is needed to get successfully installed. The activities involved in the process are coordinated by an infrastructure component which is part of the operating system and called herein setup infrastructure or abbreviated SI.

Detail Description Paragraph (6):
[0032] In this disclosure `setup` is defined as the process to automate the installation, customization and activation of software packages or individual program products.

Detail Description Paragraph (11):
[0037] `LDAP`: Lightweight Directory Access Protocol. A programming protocol adopted by the IETF (Internet Engineering Task Force) to access directories.

Detail Description Paragraph (12):
[0038] `X.500` defines the organization and working of directories according to the ISO OSI Directory Standard (CCITT Recommendation X.500). It is recommended by the International Telegraph and Telephone Consultative Committee.

Detail Description Paragraph (21):
[0047] The setup process according to the invention can be divided into four phases:

Detail Description Paragraph (27):
[0053] In a clustered environment a plurality of computer systems can be grouped together forming a system cluster which is essentially a group of individual computers working together in a coordinated fashion to run a plurality of program systems. An example of a cluster system is an OS/390 Sysplex. From a setup point of view the individual program systems are called Images. As defined above, an Image is a software entity that can be subject of an initial program load (IPL) operation and that usually comprises a plurality of program products. The bind process binds a newly installed product or package to an existing Image. This means that the physical install process is done once for the whole cluster. The Bind and Activate phase will tie the new function to the individual images of the cluster. This is done either serially initiated by the system administrator one after the other, or in one operation to all Images. During the Bind process the system administrator defines which Image to bind to.

Detail Description Paragraph (32):
[0058] 2 The Setup Infrastructure

Detail Description Paragraph (33):
[0059] A preferred implementation of the invention as shown in the general block diagram of FIG. 2 comprises a setup infrastructure 20 which is part of an operating system used to control a large computer system 21 of which in FIG. 2 only the program library disks are represented. The large computer system 21 comprises a plurality of system resources including a plurality of processors and a plurality of program packages and individual program products installed in the system and activated to run on the system. These program packages and individual program products may be utility programs such as data base administration programs, communication controller or security packages. At least some of the program packages and individual program products have interdependencies among each other. Additional program packages or individual programs are loaded into the system 21 from a program

source 22. For these additional program packages or individual programs interdependencies have to be established to at least some of the program packages and products already installed in the system 21. Three types of plug-ins 23, 24 hand 26 are connected to the setup infrastructure 20 which are used by the setup infrastructure 20 to coordinate the software install process as described below in more detail. The setup infrastructure 20 is also connected to a management directory 27 which is used to store data of a program package or program product during the install process and thereafter. The management directory 27 thus contains data of the program packages and individual program products which were previously installed and contains references to code portions of these program packages and individual program products as included in the program library of system 21.

Detail Description Paragraph (34):
[0060] The setup infrastructure 20 does not have specific knowledge on how to change and modify the system resources, nor does it know about program package formats, and it does not know in advance about program packages to install. The setup infrastructure 20 relies on the three types of plug-ins 23, 24 and 26 which provide the required information for the installation process.

Detail Description Paragraph (35):
[0061] System resources providing bind support plug-ins 23 are called SI-Supporter plug-ins or short SI-Supporters. Software packages that use the Setup Infrastructure to get installed are represented by plug-ins 24 called SI-exploiter plug-ins or short SI-exploiters. The SI-Exploiters contain a package description and product description, customization dialogs and instantiation support serving to modify these components. The plug-ins 23 and 24 represented in FIG. 2 as multiple blocks are in practice code portions which are stored in the system library of system 21 and accessed by the setup infrastructure 20 to perform the required services during the installation, customization and activation of software packages or products.

Detail Description Paragraph (36):
[0062] FIG. 3 shows a more functional representation of the setup infrastructure 20 and the related components. The setup infrastructure 20 provides generic services including install, customize, bind and activate services. An SI-Exploiter belongs to a certain program product or a certain package of program products to be installed and includes the following parts:

Detail Description Paragraph (43):
[0069] In general, there may exist many (i.e. hundreds) SI-Exploiters and much fewer (i.e. dozens) SI-Supporters. Examples of SI-Supporters are: system parameter support (PARMLIB), security packages (RACF), transaction monitors (CICS, IMS), data base administrators, workload management support (WLM), etc. These SI-Supporters help SI-Exploiters to setup the correct run-time environments.

Detail Description Paragraph (44):
[0070] Bind Requests are usually defined generically and are void of any specific supporter implementation detail. Because of this fact and because of the strict separation of exploitation and supporting roles, one can easily replace SI-Supporters with new software levels or equivalent competitive products without impacting SI-Exploiters. For example, considering bind support for directory services, an SI-Exploiter can define a new directory schema using the AddSchema bind request. For this example two different, mutually exclusive implementations of directory services are considered, one using a relational, the other a hierarchical database as backup store. Both will offer their own AddSchema bind support implementation, an exploiter will not notice the difference.

Detail Description Paragraph (47):
[0073] A third type of plug-ins which are called Package Adapters plug-ins 26 or short Package Adapters (PA) define and understand specific package formats (FIGS. 2 and 3). There exists a PA for each data format which installable program products may have. Examples of such data formats are SMPE, File Packs, TAR, or Java Jars. The Package Adapters 26 will transform their private data model to the data model defined by setup infrastructure 20 and store this transformed data into the management directory 27. The setup infrastructure 20 includes a default Package Adapter which is used for the program load operation when the data format of the program to be installed is not yet known to the setup infrastructure 20.

Detail Description Paragraph (48):
[0074] 3 The Management Directory

Detail Description Paragraph (49):
[0075] An important part of the system described is the management directory 27 which is the storage of data and references as used by the setup infrastructure 20 for performing the install process. The management directory 27 may be an X.500 based directory that is accessed by using the LDAP protocol. The Data Model used by directory 27 is a CIM compliant data model proposed by DMTF. This data model is defined as an hierarchical structure which enables grouping of related data. For instance, all data that is associated with a particular software package or product is defined as subordinate data.

Detail Description Paragraph (68):
[0094] The product definition will be read during the install phase and placed into the management directory 27. During the remaining setup phases, the product definition will be augmented with additional data and additional entries will be added to the definitions in the management directory 27.

Detail Description Paragraph (72):
[0098] In this case the actual value will be determined at bind time. The reason for this is to enable detection of users of values. In case the administrator has to change the actual dataset name, the setup infrastructure 20 can determine which other entries are depending on that value and, in such case, re-schedule the bind processing.

Detail Description Paragraph (74):
[0100] As shown in the functional flow diagram of FIG. 4 the setup process 40 is subdivided into the four major phases install 41, customize 42, bind 43 and activate 44 which are invoked separately. At any point in time the effects of one phase can be undone.

Detail Description Paragraph (76):
[0102] The install phase 41 selects the appropriate Package Adapter 45 which will process the product definitions 46 and place them in the management directory 27. At this point, the setup infrastructure 20 knows all about the program product which may be part of a program package, and it can use a standard dialog to determine where the libraries of the program product should be placed in the system 21. The standard dialog should also be able to determine that all *install* requisites are able to be satisfied and to drive the appropriate installation program. During this phase the setup infrastructure 20 will register where to find the customization dialogs and the product instantiation program and whether the program product introduces a new SI-Supporter 23.

Detail Description Paragraph (89):
[0115] Herein `Status` indicates the Setup status of a program product.

Detail Description Paragraph (91):
[0117] During the customize phase 42 the setup infrastructure 20 invokes the customization dialogs 47 of the selected products via the management directory 27 from the system 21. The dialogs are written using a specialized XML vocabulary. The dialogs will offer the administrator the possibility to modify default settings and will then generate appropriate bind requests which are placed in the management directory 27 as well.

Detail Description Paragraph (92):
[0118] Because of the use of the management directory 27 the amount of information required to be specified by the human administrator of the system has been reduced. For example, DB2 customization currently requires the data set name for Language Environment libraries. After the install phase is extending to the Language Environment, this information is available in the management directory 27 and does not need to be requested by the administrator. Other information, like an IP HOSTNAME, can now automatically be satisfied, thereby further reducing the input required during customization.

Detail Description Paragraph (108):
[0134] As described before, binding is the process of tying a program package or product to an image. After the system administrator has indicated which image to bind to by setting up the appropriate bind requests, setup infrastructure 20 will examine these bind requests and invoke the corresponding bind service routines 48 in the management directory 27. The invoked bind service routines 48 will examine the

bind requests and perform the requested functions.

Detail Description Paragraph (111):
[0137] FIG. 5 shows the data flow of the bind request processing in form of the transfer of data to and from the management directory 27. During the customization dialog 51 a Job name which, for example, may be BBOSR and belongs to a WEB Server 50, is registered into the management directory 27 among the customization properties 52 of the product. As part of registering the product definitions, corresponding the SI-Exploiters produce Bind-Requests which are also stored in the management directory 27. The Bind-Requests may refer to at least one SI supporter 54 which are invoked by the setup infrastructure 20 to be registered into the management directory 27. By means of the product definitions bind support 55 indicates that the product offers bind support for other program products. In the example shown, there are SI-Supporters registered in the directory 27 for database, security, networking and WEB application packages. These SI-Supporters are made available to the product instantiation 59 and to the bind/unbind services 58.

Detail Description Paragraph (129):
[0155] As the last step, the system administrator can ask setup infrastructure 20 to activate the WebServer of the above example. SI 20 will process the activate request as added to the management directory by the bind phase. The updated product definitions of the example will then look as follows:

Detail Description Paragraph (142):
[0168] Although the bind request and the activation request have been executed, all data associated with the product are kept in the management directory 27 and not removed.

Detail Description Paragraph (145):
[0171] Policy definitions are placed in the management directory 27 such as, for example:

Detail Description Paragraph (152):
[0178] Because all details about program products are kept in the management directory 27, removal of program products from the system 21 is fully supported. The phases described above, except customization, will be executed in reverse.

Detail Description Paragraph (158):
[0184] Since all data pertaining to products is kept in the management directory 27, the setup infrastructure 20 is able to export a completely customized program product or collections of program products. The setup infrastructure 20 will produce sets of files and the updated product description which contains all customization data. These sets of files can be shipped to other locations or systems. The export function will produce a software package that can be processed by the default package adapter 26. Today, customers use various techniques to deploy software packages throughout their enterprises. The export function of the system and method according to the invention provides a structured means for customers to:

CLAIMS:

1. A system for request based installation, customization and activation of software packages or products on a computer system operated by a system control program which provides support for the installation of computer program packages or products, said computer system having a plurality of system resources including a plurality of program packages and products stored in the system, where at least some of said program packages and products have interdependencies among each other and where for said program packages or products to be installed interdependencies have to be established to at least some of the program packages and products stored, the system comprising: setup infrastructure means for coordinating the loading, customization, binding and activation of the program package or product to be installed; a plurality of first plug-in means each connected to one of said system resources and used by the setup infrastructure means for creating a product definition, for providing customization dialogs and for support to initialize and modify said program package or product; a plurality of second plug-in means each connected to selected ones of said system resources for providing bind services to the setup infrastructure means, said bind services are used by said first plug-in means to establish binding to at least some of the program products already installed, where the binding parameters needed by the first plug-in means are indicated to the second plug-in means by Bind Requests.

2. A system according to claim 1, comprising a plurality of plug-in means of a third type which identify various data formats, one of said third type plug-ins is assigned to a program product to be installed for indicating to the setup infrastructure means the specific package formats as used by said program.

5. A system according to claim 2, wherein the third type plug-in means comprises means for transforming a private data model of the program product to the data model defined by the setup infrastructure means.

6. A system according to claim 1, further comprising a management directory connected to the setup infrastructure means for storing all data associated with a particular software package or product and references to portions of said software package or product in a system library, said references are used by the setup infrastructure means for accessing data of a software package or product in a system library wherein said software package or product is stored.

7. A system according to claim 6, wherein the management directory comprises a data model which is defined as an hierarchical structure and which enables grouping of related data.

8. A system according to claim 6, comprising management policy definitions how bind requests are to be processed, said policy definitions are stored in the management directory and are used by the setup infrastructure means to block or modify bind requests until approval has been given.

12. A method according to claim 9, further comprising (f) storing all data associated with a particular software package or product in a management directory, data associated with a particular software package or product and including references to portions of said software package or product in a system library; and using said references for accessing said data in the system library.

13. A method according to claim 12, comprising defining the data model of the management directory as an hierarchical structure which enables grouping of related data.

14. A method according to claim 12, comprising defining policy rules how bind requests are to be processed, storing said policy rules in the management directory and using said policy rules in the coordinating operation of step (a) to block or modify bind requests until approval has been given.

15. A method according to claim 9, comprising using the data and references stored in the management directory for removal of program products from the system.

16. A method according to claim 9, comprising using the data and references stored in the management directory for exporting completely customized program products or collections of program products to other locations.